Differential Secrecy for Distributed Data and Applications to Robust Differentially Secure Vector Summation

Kunal Talwar
Apple, Cupertino, CA, USA

Abstract

Computing the noisy sum of real-valued vectors is an important primitive in differentially private learning and statistics. In private federated learning applications, these vectors are held by client devices, leading to a distributed summation problem. Standard Secure Multiparty Computation protocols for this problem are susceptible to poisoning attacks, where a client may have a large influence on the sum, without being detected.

In this work, we propose a poisoning-robust private summation protocol in the multiple-server setting, recently studied in PRIO [14]. We present a protocol for vector summation that verifies that the Euclidean norm of each contribution is approximately bounded. We show that by relaxing the security constraint in SMC to a differential privacy like guarantee, one can improve over PRIO in terms of communication requirements as well as the client-side computation. Unlike SMC algorithms that inevitably cast integers to elements of a large finite field, our algorithms work over integers/reals, which may allow for additional efficiencies.

2012 ACM Subject Classification Security and privacy → Privacy-preserving protocols

Keywords and phrases Zero Knowledge, Secure Summation, Differential Privacy

Digital Object Identifier 10.4230/LIPIcs.FORC.2022.7

Acknowledgements We would like to thank Ulfar Erlingsson for many helpful discussions, and the anonymous referees for their feedback.

1 Introduction

We investigate the problem of distributed private summation of a set of real vectors, each of Euclidean norm at most 1. Each client device holds one of these vectors and the goal is to allow a server to compute the sum of these vectors. Privacy constraints require that an adversary not learn too much about any of these vectors, and this constraint will be expressed as a differential privacy [16] requirement.

This is a common primitive to private federated learning and statistics. In a setting of a trusted server, the clients could send the vectors to the server, which could then output the sum with appropriate noise added to ensure differential privacy. A natural solution then is to use tools from secure multiparty computation to simulate this trusted server. This approach goes back to the early days of differential privacy [15], and has been heavily investigated [11, 8]. Practical protocols applying this approach have to deal with clients dropping out during the protocol, and often scale poorly with the number of clients. The security guarantee of SMC ensures that we learn nothing except the (noisy) sum. However, a malicious client in many of these protocols can contribute a vector with arbitrarily large norm and go completely undetected. Addressing this manipulability would require additional modification to these protocols, making them less feasible.

An elegant way out is possible under slightly stronger trust assumptions. Corrigan-Gibbs and Boneh [14] show that if we have a set of S servers where at least one of them is trusted, we can efficiently get both privacy and integrity[1]. In our application, this framework gives a protocol that validates that each vector has norm at most 1, and computes the sum of vectors. The security guarantee here says that other than the output and the fact that the inputs have norm at most 1, any strict subset of servers learns nothing about the clients' inputs. If the clients add a small amount of noise to their inputs, or more generally, use a local randomizer, the final output can be shown to be differentially private. As long as all the inputs are bounded in norm, the validity predicates are all 1 and hence have no information. The overall security guarantee then says that the view of any strict subset of servers is differentially private with respect to the input vectors.

Note that perfect secrecy here is impossible as the output itself leaks information about the inputs. In the approach described above, *What we compute* does not leak too much about the input since we are computing a differentially private output. *How we compute it*, i.e. the computation protocol itself provides perfect secrecy *subject to* the output.

Our guarantee of interest is the leakage about any input from the process as well as the output, i.e. the sum of the privacy costs from the *what* and the *how*. In this work, we relax the secrecy guarantee of the protocol to a differential secrecy guarantee. We show that this allows for simpler and more efficient algorithms for the robust vector summation problem.

As a warm-up, we first show a natural variant of secret sharing that satisfies differential secrecy. We next show that one can privately verify that theof a secret-shared vector is bounded, if one allows some slack. We present a simple protocol based on random projections. Our protocol accepts all vectors of norm at most 1 with high probability. Additionally, a vector with too large a norm (polylogarithmic in the parameters) will be rejected with high probability. Thus we have some robustness: a malicious client can affect the sum by more than norm 1, but not arbitrarily more. Our privacy proof here relies on a new result on the privacy bounds for noisy random projections. Unusually for a differential privacy result, here we exploit the randomness of the "query". Compared to PRIO, our verification algorithm requires no additional work from clients, and requires less communication between servers.

With secret-sharing and norm-verification over secret shares in place, our algorithm for summation is simple. The clients secret-share their vectors, and the servers run the norm-verification protocol on all the clients. For the clients that pass the norm verification, each server adds up their secret shares. The servers now hold additive secret shares of the summation, which can be communicated between servers to derive the vector summation.

This then eliminates the need for the client to perform any additional computation ($\Theta(d)$ in PRIO) or communication ($\Theta(\sqrt{d})$ [9] in PRIO). The validity check comes at zero cost to the client. This comes at a small increase in the inter-server communication from 3 field elements to a logarithmic number of real numbers.

Our algorithms can work over real numbers or integers, instead of finite fields. Compressing these to reduce communication, for example by truncating or rounding does not affect the privacy guarantee, allowing one to find a representation that provides an acceptable tradeoff between accuracy and communication cost.

In practice, as we discuss in Section 7, this can be a significant saving, especially in settings such as federated learning where the vectors being aggregated are high-dimensional gradients and the client to server communication is often the bottleneck. For typical parameters, where PRIO would need a large finite field needing 128 bits per coordinate (or at the very least 32 bits per coordinate), using real numbers can bring us down to 8 or 16 bits per coordinate.

[1] We defer the precise definitions to Section 3

Several natural questions remain. Our norm verification, and hence our robustness guarantee for summation, is approximate. We reject vectors with large enough norm. It would be interesting to reduce, or even eliminate this approximation, while maintaining the efficiency advantages of our protocol. Given the practical relevance of robust summation, it would also be compelling to improve distributed proofs of norm bound in the standard PRIO setting.

Finally, relaxing perfect secrecy in secure multiparty computation, or more broadly in cryptography to differential secrecy may allow for more efficient protocols in other settings.

2 Related Work

The question of simultaneously studying the differentially private function (the *What*) and the cryptographic protocol for computing it (the *How*) was first studied by Beimel, Nissim and Omri [5]. They showed that in the Secure Function Evaluation (SFE) setting without a trusted server, one can provably gain in efficiency of the protocol for summing 0-1 values. This differential privacy-based definition of security was subsequently used by Backes et al. [2], who show that this relaxation allows one to use imperfect randomness in certain cryptographic protocols.

Private anonymous summation protocols using mutliple servers go back to at least the split-and-mix protocol of Ishai et al. [23]. In the context of differential privacy, these have gained a lot of importance given recent results in the shuffle model of privacy [7, 17, 13, 4]. Recent works [3, 20, 19] have improved the efficiency of these results. These protocols however suffer from the manipulability issue: it is easy for one malicious client to significantly poison the sum without getting detected.

Another line of work [8] proposes practical secure summation protocol under different trust assumptions. These protocols also suffer from the manipulability problem. Recent works such as [29, 6] address the scaling challenges in that work.

The two-party version of some of these questions have been studied by [27, 22]. Kairouz, Oh and Vishwanathan [25, 24] study private secure multiparty computation under a local differential privacy constraint. In a different vein, Cheu, Smith and Ullman [12] show that locally differentially private algorithms are fairly manipulable by small subsets of users, and quantify their manipulability.

3 Definitions

We would like the protocol to satisfy several properties. We define appropriate notions of these first.

▶ **Definition 1** (Completeness). *A protocol* Π *is* $(1-\beta)$*-complete w.r.t.* $\mathcal{L}$ *if for all* $x \in \mathcal{L}$*, the protocol accepts* x *with probability at least* $(1-\beta)$*.*

▶ **Definition 2** (Soundness). *A protocol* Π *is* β*-sound w.r.t.* $\mathcal{L}$ *if for* $x \notin \mathcal{L}$*, the protocol accepts with probability at most* β*.*

Let $\mathcal{L}_r$ denote the set of vectors with $\|\cdot\|_2$ norm at most r. We will show completeness w.r.t. $\mathcal{L}_1$ and soundness w.r.t. $\mathcal{L}_\rho$. for a parameter $\rho > 1$.

Additionally, we would like a mild relaxation of Zero Knowledge, inspired and motivated by the notion of Differential Privacy. We first recall a notion of near-indistinguishability used in Differential Privacy:

▶ **Definition 3.** *Two random variables P and Q are said to be (ε, δ)-close, denoted by $P \approx_{(\varepsilon,\delta)} Q$ if for all events S, $Pr[P \in S] \leq \exp(\varepsilon) \cdot \Pr[Q \in S] + \delta$, and similarly, $Pr[Q \in S] \leq \exp(\varepsilon) \cdot \Pr[P \in S] + \delta$*

One can relax the secrecy requirements in cryptography to differential secrecy. Here we define this notion for Zero Knowledge[2].

▶ **Definition 4.** *We say a protocol Π is (ε, δ)-Differentially Zero Knowledge w.r.t. $\mathcal{L}$ if there is a distribution Q such that for all $x \in \mathcal{L}$, the distribution $\Pi(x)$ of the protocol's transcript on input x satisfies $\Pi(x) \approx_{(\varepsilon,\delta)} Q$.*

Note that here we require privacy, or differential zero knowledge for $x \in \mathcal{L}$. While one can naturally define a computational version of this definition, along the lines of computational differential privacy definitions [28], we restrict ourselves to the information-theoretic version in this work.

In this work, we will be using multi-verifier protocols. Here the notion of near Zero Knowledge is with respect to a strict subset of verifiers. The definition here is the simplest one where the prover starts with an input x and verifiers start with no input, as this will suffice for our purpose. It can naturally be extended to other setups, for example when the verifiers already hold some shares of the input and a witness.

▶ **Definition 5.** *A single-prover, multiple-verifier protocol Π is (ε, δ)-Differentially Zero Knowledge w.r.t to a subset T of parties if there is a distribution Q dependent only on inputs of T and the output of the protocol, such that for any input $x \in \mathcal{L}$, the distribution of messages from T^c to T is (ε, δ)-close to Q.*

Attack Models. In our work, the client will play the role of the prover, and the servers will play the role of the verifiers. We interchangeably use client/server and prover/verifier terminology as appropriate. We will prove completeness and privacy for honest-but-curious prover. We will establish soundness against an arbitrary malicious provers. This implies that a client that is behaving according to the protocol will get a strong privacy guarantee, and will be accepted with high probability. A malicious client will still likely be caught, and may not get a privacy assurance. Our protocols will have privacy against an a strict subset of servers being malicious, as long as at least one of the servers is honest. The soundness and completeness results will assume that all servers are honest. Thus some subsets of servers behaving maliciously can hurt the utility of the protocol, but not the privacy.

For the robust aggregation problem, we argue the privacy of the process given the final sum, and leave to a different argument the question of the privacy of the sum itself. The protocol can easily be modified by adding noise to the sum to ensure that the sum itself is private; this can be done by having each server add sufficient noise, or by implementing a distributed noise-addition protocol. Our modular approach allows us to separately analyze the privacy cost of the output of the protocol. In particular, we may apply different analyses depending on whether we consider distributed noise addition, or apply local randomizers and rely on privacy amplification by shuffling. We defer additional discussion to Section 7.

[2] This is the *local DP* version of ZK which is appropriate in this setting. One can similarly define a central DP version, where the simulator has access to all but one client's input

Secure Summation

The secure summation problem is defined as follows. There is a set of N clients with client i holding a vector $\mathbf{x}_i \in \mathbb{R}^d$ with $\|\mathbf{x}_i\| \leq 1$. Our goal is to design a protocol with S servers such that for suitable parameters $\varepsilon, \delta, \rho, \beta$, the following properties hold:

Correctness: When all parties are honest, the protocol allows a designated server to compute a vector $\mathbf{y} \in \mathbb{R}^d$ such that $\mathbf{y} = \sum_i \mathbf{x}_i$ with probability at least $(1-\beta)$.

Privacy: For any honest client i, the protocol is (ε, δ)-Differentially Zero Knowledge w.r.t. any subset of parties that excludes at least one server.

Robustness: For any possibly malicious client i, the computed summation $\mathbf{y}$ differs from the output $\mathbf{y}_{-i}$ without client i in norm by at most ρ, i.e. $\|\mathbf{y} - \mathbf{y}_{-i}\|_2 \leq \rho$, except with probability at most β.

In words, we would like a protocol that is private w.r.t. to any honest client as long as at least one of the S servers is honest. Thus an honest client that trusts at least one of the servers to be honest is assured of a differential privacy guarantee. The robustness property gives an integrity guarantee if all servers are honest. The parameter $\rho \geq 1$ controls how much any client can impact the output of the protocol. Note that a malicious client can always behave as if their input was $\mathbf{x}'_i$ for any arbitrary vector of norm 1. The robustness requirement here puts an upper bound on how much a malicious client can distort the summation The correctness and robustness properties will allow failure with probability β. Depending on the application, a small constant β may be acceptable.

4 Preliminaries

We state two important properties of the differential privacy notion of closeness.

▶ **Proposition 6.** *Suppose that* $P \approx_{(\varepsilon,\delta)} Q$ *and* $P' \approx_{(\varepsilon',\delta')} Q'$. *Then*

Post Processing: *For any function* f, $f(P) \approx_{(\varepsilon,\delta)} f(Q)$.

Simple Composition: $(P, P') \approx_{(\varepsilon+\varepsilon',\delta+\delta')} (Q, Q')$.

The following is a restatement of the privacy of the Gaussian mechanism [16, Thm A.1].

▶ **Lemma 7.** *Let* $\varepsilon, \delta > 0$ *and let* $\mathbf{x} \in \mathbb{R}^d$ *satisfy* $\|\mathbf{x}\|_2 \leq 1$. *Let* $P \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{I}_d)$ *and let* $Q \sim \mathbf{x} + \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{I}_d)$. *Then* $P \approx_{(\varepsilon,\delta)} Q$ *if* $\sigma \geq 2\sqrt{\ln \frac{2}{\delta}}/\varepsilon$.

We next prove the following simple result on the privacy properties of noisy random projections.

▶ **Lemma 8.** *Let* G *be a random matrix in* $\mathbb{R}^{k \times d}$ *such that for a constant* c_δ, *every* $\mathbf{x} \in \mathbb{R}^d, \|\mathbf{x}\| \leq 1$ *satisfies*

$$\Pr[\|G\mathbf{x}\| \geq c_\delta] \leq \delta, \tag{1}$$

where the probability is taken over the distribution of G. *Let* $\sigma = 2c_\delta\sqrt{\ln \frac{2}{\delta}}/\varepsilon$. *Then for any* $\mathbf{x} \in \mathbb{R}^d$ *with* $\mathbf{x} \leq 1$,

$$(G, \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{I}_k)) \approx_{(\varepsilon,2\delta)} (G, G\mathbf{x} + \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{I}_k)).$$

Proof. Fix $\mathbf{x}$ and let $\mathcal{E}$ be the event that $\|G\mathbf{x}\| \geq c_\delta$. By Lemma 7, we have that conditioned on the event $\mathcal{E}$,

$$(G, \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{I}_k)) \approx_{(\varepsilon,\delta)} (G, G\mathbf{x} + \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{I}_k)).$$

By Equation (1), $\Pr[\mathcal{E}] \geq 1-\delta$. The claim now follow from the definition of (ε, δ)-closeness. ◀

We next recall a version of the Johnson-Lindenstrauss lemma on the length of random projections.

▶ **Lemma 9** (Gaussian Ensemble JL). *Let $G \in \mathbb{R}^{k\times d}$ be a random matrix where each $G_{ij} \sim \mathcal{N}(0, \frac{1}{k})$. Then for any $\mathbf{x} \in \mathbb{R}^d$ with $\|\mathbf{x}\| \le 1$,*

$$\Pr[\|G\mathbf{x}\| \notin (1 \pm O(\sqrt{(\ln \tfrac{1}{\delta})/k))\|\mathbf{x}\|] \le \delta$$

To get more precise estimates, we recall that the sum of squares of k $\mathcal{N}(0, \frac{1}{k})$ random variables is distributed as a (scaled version of a) chi-square distribution χ^2_k. We will use the following tail bounds for χ^2_k random variables from Laurent and Massart [26, Lemma 1 rephrased]:

▶ **Theorem 10.** *Let Q be a χ^2_k random variable. Then for any $\beta > 0$,*

$$\Pr[\frac{1}{k}Q \le 1 - 2\sqrt{x/k}] \le \exp(-x),$$
$$\Pr[\frac{1}{k}Q \ge 1 + 2\sqrt{x/k} + 2x/k] \le \exp(-x).$$

Combining Theorem 10 with Lemma 8, we get the following useful corollary.

▶ **Corollary 11.** *Let $G \in \mathbb{R}^{k\times d}$ be a random matrix where each $G_{ij} \sim \mathcal{N}(0, \frac{1}{k})$ and let $c_\delta = \sqrt{1 + 2\sqrt{(\ln \frac{1}{\delta})/k} + 2(\ln \frac{1}{\delta})/k}$. Let $\sigma = 2c_\delta\sqrt{\ln \frac{2}{\delta}}/\varepsilon$. Then for any $\mathbf{x} \in \mathbb{R}^d$ with $\mathbf{x} \le 1$,*

$$(G, \mathcal{N}(\mathbf{0}, \sigma^2\mathbb{I}_k)) \approx_{(\varepsilon,2\delta)} (G, G\mathbf{x} + \mathcal{N}(\mathbf{0}, \sigma^2\mathbb{I}_k)).$$

5 Warm-up: Secret Sharing Real-valued Vectors

As a prelude to our result on norm verification, we first show how the standard secret sharing protocol extends to real-valued vectors, when allowing for Differential secrecy. Consider the protocol for secret-sharing a real-valued vector of norm at most 1 between S servers shown in Algorithm 1.

Algorithm 1 Secret Sharing a real vector.

Prover($\mathbf{x}$):
  Input: Vector $\mathbf{x} \in \mathbb{R}^d$ with $\|\mathbf{x}\| \le 1$.
  Parameters: $\sigma_{SS} \in \mathbb{R}$.
  Generate $\mathbf{g}_1, \ldots, \mathbf{g}_{S-1} \sim \mathcal{N}(\mathbf{0}, \sigma^2_{SS}\mathbb{I}_d)$ using private randomness.
  Send $\mathbf{x} - \sum_{i=1}^{S-1} \mathbf{g}_i$ to Verifier 0.
  for $i = 1 \ldots S-1$ **do**
    Send $\mathbf{g}_i$ to Verifier i.

To prove the differential secrecy for this protocol, we show a simulator for any subset of verifiers in Algorithm 2.

We next argue that this secret sharing scheme is differentially secure.

▶ **Theorem 12.** *Fix any $T \subsetneq [S]$. Then Prover($\mathbf{x}$)$|_T \approx_{(\varepsilon,\delta)}$ Simulator(T) for $(S - |T|)\sigma^2_{SS} \ge 4\ln \frac{2}{\delta}/\varepsilon^2$.*

Algorithm 2 Simulator for Algorithm 1.

Simulator($T \subsetneq [S]$):
  Input: T proper subset of S
  Parameters: $\sigma_{SS} \in \mathbb{R}$.
  for $i \in T$ **do**
    if $i \neq 0$ **then**
      Generate $\mathbf{g}_i \sim \mathcal{N}(\mathbf{0}, \sigma_{SS}^2 \mathbb{I}_d)$.
      Send $\mathbf{g}_i$ to Verifier i.
  if $0 \in T$ **then**
    Generate $\mathbf{g} \sim \mathcal{N}(\mathbf{0}, (S - |T|)\sigma_{SS}^2 \mathbb{I}_d)$.
    Send $\mathbf{g} - \sum_{i \in T; i \neq 1} \mathbf{g}_i$ to Verifier 0.

Proof. If $0 \notin T$, the simulation is perfect: indeed each verifier in T receives an independent Gaussian vector with variance $\sigma_{SS}^2 \mathbb{I}_d$ in both distributions. When $0 \in T$, consider the distribution of the message to Verifier 0 conditioned on $T \setminus [0]$.

The simulator output to Verifier 0 is distributed as $\mathcal{N}(-\sum_{i \in T; i \neq -0} \mathbf{g}_i, (S - |T|)\sigma_{SS}^2 \mathbb{I}_d)$. The message to verifier 0 from the prover, conditioned on $\{\mathbf{g}_i\}_{i \in T: i \neq 0}$ is distributed as $\mathcal{N}(\mathbf{x} - \sum_{i \in T; i \neq 0} \mathbf{g}_i, (S - |T|)\sigma_{SS}^2 \mathbb{I}_d)$. The claim now follows from the privacy of the Gaussian mechanism (Lemma 7). ◀

The differential secrecy implies that an honest prover's privacy is protected against an arbitrary collusion of verifiers short of all of them. Note also that by making σ_{SS} larger, we can improve the privacy cost. A larger σ_{SS} only costs us in terms of the precision to which these messages should be communicated to ensure that the sum of secret shares is close to $\mathbf{x}$. Note that we can post-process these vectors (both in the algorithm and its simulation), e.g. by rounding or truncation. By the post-processing property of differential privacy, the differential secrecy is maintained.

6 Differential Zero Knowledge Proofs of bounded norm

We next describe our DZK protocol to verify a Euclidean norm bound. The first step is to secret-share the vector between the two verifiers as in the previous section. The rest of the protocol only involves the verifiers; the prover code therefore is identical to secret-sharing.

The second step is norm estimation and happens amongst the verifiers. As a first cut, suppose that the servers aggregate their shares, while adding noise to each share to preserve privacy. This would require adding d-dimensional gaussian noise to each share. This noise being fresh and independent will contribute to the norm of the computed sum, which will now be about $\sqrt{d}$, and will have variance growing polynomially with d. This will make it impossible to estimate the norm better than some polynomial in d, and thus our gap ρ will grow polynomially with the dimension.

To improve on this, we will use random projection into a k-dimensional space for a parameter k independent of the dimension. Being a lower-dimensional object, a projection can be privately estimated much more accurately. The choice of the projection dimension k will give us a trade-off between the privacy parameters and the gap assumption. Intuitively, we rely on the Johnson-Lindenstrauss lemma, which says that the Euclidean norm of a vector is approximately preserved under random projections. Since projection is a linear operator, computing the projection of a secret-shared vector is straight-forward. Verifier 0 here takes the special role of collecting an estimate of a random projection of $\mathbf{x}$, computing its norm and sharing the Accept/Reject bit.

Algorithm 3 Protocol for Norm Verification.

Input: Prover has a vector $\mathbf{x} \in \mathbb{R}^d$
Output: Verifiers must agree on Accept.

Prover($\mathbf{x}$*)*:
- **Input:** Vector $\mathbf{x} \in \mathbb{R}^d$ with $\|\mathbf{x}\| \leq 1$.
- **Parameters:** $\sigma_{SS} \in \mathbb{R}$.
- Generate $\mathbf{g}_1, \ldots, \mathbf{g}_{S-1} \sim \mathcal{N}(\mathbf{0}, \sigma_{SS}^2 \mathbb{I}_d)$ using private randomness.
- Send $\mathbf{x} - \sum_{i=1}^{S-1} \mathbf{g}_i$ to Verifier 0.
- **for** $i = 1 \ldots S-1$ **do**
  - Send $\mathbf{g}_i$ to Verifier i.

Verifier-0:
- **Parameters:** Integer k. Threshold $\tau \in \mathbb{R}$.
- Receive $\mathbf{z}_0$ from Prover. `// Expected to be` $\mathbf{x} - \sum_{i=1}^{S-1} \mathbf{g}_i$
- Generate $\mathbf{W} \in \mathbb{R}^{k \times d}$ with each $W_{ij} \sim \mathcal{N}(0, \frac{1}{k})$ using private randomness.
  `// This version assumes honest Verifier 0. To allow malicious Verifier 0,` W `is generated using randomness shared amongst verifiers.`
- Send $\mathbf{W}$ to Verifiers $1, \ldots, S-1$.
- **for** $i = 1 \ldots S-1$ **do**
  - Receive $\mathbf{y}_i$ from Verifier i. `// Expect` $\mathbf{y}_i = \mathbf{W}\mathbf{g}_i + \mathit{Noise}$.
- Compute $\mathbf{v} = \mathbf{W}\mathbf{z}_0 + \sum_{i=1}^{S-1} \mathbf{y}_i + \mathcal{N}(0, \sigma_v^2 \mathbb{I}_k)$. `// Expect` $\mathbf{v} = \mathbf{W}\mathbf{x} + \mathit{Noise}$.
- **if** $|\mathbf{v}| \geq \tau$ **then**
  - Accept $= 0$
- **else**
  - Accept $= 1$
- Send Accept to Verifiers $1, \ldots, S-1$.

Verifier-i $(i \geq 1)$:
- **Parameters:** Integer k. Noise scale $\sigma_v \in \mathbb{R}$.
- Receive $\mathbf{z}_i$ from Prover. `// Expected to be` $\mathbf{g}_i$
- Receive $\mathbf{W} \in \mathbb{R}^{k \times d}$ from Verifier-0.
- Compute $\mathbf{y}_i = \mathbf{W}\mathbf{z}_i + \mathcal{N}(0, \sigma_v^2 \mathbb{I}_k)$.
- Send $\mathbf{y}_i$ to Verifier-0.
- Receive Accept from Verifier-0.

We start with establishing compeleteness, which will determine the acceptance threshold τ. We will then show soundness for an appropriate ρ.

▶ **Theorem 13** (Completeness). *Suppose that the prover and the verifiers are honest and the* $\|\mathbf{x}\| \leq 1$. *Then for* $\tau \geq \sqrt{(\frac{1}{k} + |S|\sigma_v^2)(k + 2\ln\frac{1}{\beta} + 2\sqrt{k \ln \frac{1}{\beta}})}$,

$$\Pr[\text{Accept} = 1] \geq 1 - \beta.$$

Proof. Under the assumptions, $W\mathbf{x}$ is distributed as $\mathcal{N}(0, \frac{\|\mathbf{x}\|_2^2}{k}\mathbb{I})$. The noise added by each server is distributed as $\mathcal{N}(0, \sigma_v^2 \mathbb{I})$, and all of these Gaussian random variables are independent. Thus $\mathbf{v}$ computed by Verifier 0 is distributed as $\mathcal{N}(0, (\frac{\|\mathbf{x}\|_2^2}{k} + |S|\sigma_v^2)\mathbb{I})$, and its squared norm is distributed as $(\frac{\|\mathbf{x}\|_2^2}{k} + |S|\sigma_v^2)Q$, where Q is a χ_k^2 random variable. Thus

$$\Pr[\|\mathbf{v}\|_2^2 \geq \tau^2] = \Pr[Q \geq (\frac{1}{k} + |S|\sigma_v^2)^{-1}\tau^2].$$

Plugging the upper tail bounds from Theorem 10, the result follows. ◀

▶ **Theorem 14** (Soundness). *Suppose that the verifiers are honest and suppose that* $\|\sum_{i=0}^{S-1} z_i\| \geq \rho$, *where* z_i *is the message to verifier* i. *Then for* $\rho^2 \geq \frac{k\tau^2}{k-2\sqrt{k\ln\frac{1}{\beta}}} - k|S|\sigma_v^2$,

$$\Pr[\mathsf{Accept} = 1] \leq \beta.$$

Proof. As in the proof of Theorem 13, now $\|\mathbf{v}\|_2^2$ is distributed as $(\frac{\rho^2}{k} + |S|\sigma_v^2)Q$ for a χ_k^2 random variable Q. Using the lower tail bounds from Theorem 10, it suffices to ensure

$$(\frac{\rho^2}{k} + |S|\sigma_v^2)(k - 2\sqrt{k\ln\frac{1}{\beta}}) \geq \tau^2.$$

Rearranging, the claim follows. ◀

Some discussion on k is in order. A small k ensures that we need to add less noise and thus get better estimates. At the same time, larger k ensures stronger concentration of the χ_k^2 random variable. For intuition, we next estimate the bound on ρ^2 from Theorem 14, plugging in τ from Theorem 13. Setting $\lambda = \frac{\sqrt{\ln\frac{1}{\beta}}}{k}$ and assuming λ is small enough, we can write

$$\begin{aligned}
\rho^2 &= \frac{k\tau^2}{k - 2\sqrt{k\ln\frac{1}{\beta}}} - k|S|\sigma_v^2 \\
&= (1 + k|S|\sigma_v^2)\frac{k + 2\ln\frac{1}{\beta} + 2\sqrt{k\ln\frac{1}{\beta}}}{k - 2\sqrt{k\ln\frac{1}{\beta}}} - k|S|\sigma_v^2 \\
&= (1 + k|S|\sigma_v^2)\frac{1 + 2\lambda + 2\sqrt{\lambda}}{1 - 2\sqrt{\lambda}} - k|S|\sigma_v^2 \\
&\approx (1 + k|S|\sigma_v^2)(1 + O(\sqrt{\lambda})) - k|S|\sigma_v^2 \\
&= 1 + O(k|S|\sigma_v^2\sqrt{\lambda}) \\
&\approx 1 + O(|S|\sigma_v^2 k^{\frac{1}{2}}(\ln\frac{1}{\beta})^{\frac{1}{4}}).
\end{aligned}$$

Taking $k = \Theta(\sqrt{\ln\frac{1}{\beta}})$ suffices to ensure λ is small enough for the approximations above to be valid. This leads to $\rho^2 = \Theta(|S|\sigma_v^2\sqrt{\ln\frac{1}{\beta}})$. In practice, one may want to use the exact cdf for the χ_k^2 distribution instead of the tail bounds used in the theorems.

We now prove the differential zero knowledge property of the algorithm. We assume that verifier 0 is honest. We will then relax this assumption using shared randomness.

▶ **Theorem 15** (DZK assuming honest Verifier 0). *Suppose that* $\|\mathbf{x}\|_2 \leq 1$. *If the prover and Verifier-0 are honest, then for any* $T \subset [S] \setminus \{0\}$, T*'s view is* (ε, δ)*-DZK as long as* $\sigma_v \geq 2c_\delta\sqrt{\ln\frac{4}{\delta}}/\varepsilon$.

Proof. The simulator is defined in Algorithm 4. The simulator sends messages to verifiers in T in steps 4, 6, and 13. The messages in steps 4 and 6 follows exactly the same distribution as that in the mechanism, with all $\mathbf{g}_i$'s and the matrix $\mathbf{W}$ being independent normal. The message in step 13 is the Accept bit, which is computed as a post-processing of the vector

Algorithm 4 Simulator for Algorithm 3.

1 *Simulator*($T \subsetneq [S]; 0 \notin T$)**:**
  Input: T proper subset of S
  Parameters: $\sigma_{SS}, \sigma_v, \tau \in \mathbb{R}$, integer k.
2 **for** $i \in T$ **do**
3 Generate $\mathbf{g}_i \sim \mathcal{N}(\mathbf{0}, \sigma_{SS}^2 \mathbb{I}_d)$.
4 Send $\mathbf{g}_i$ to Verifier i.
5 Generate $\mathbf{W} \in \mathbb{R}^{k \times d}$ with each $W_{ij} \sim \mathcal{N}(0, \frac{1}{k})$.
6 Send $\mathbf{W}$ to each Verifier in T.
7 Receive $\{\mathbf{y}_i\}_{i \in T}$.
8 Compute $\mathbf{v}_{Sim} = \sum_{i \in T}(\mathbf{y}_i - \mathbf{W}\mathbf{g}_i) + \mathcal{N}(\mathbf{0}, (S - |T|)\sigma_v^2 \mathbb{I}_k)$.
9 **if** $|\mathbf{v}_{Sim}| \geq \tau$ **then**
10 Accept $= 0$
11 **else**
12 Accept $= 1$
13 Send Accept to all verifiers.

$\mathbf{v}_{Sim}$ computed in step 8. The corresponding Accept bit in the protocol is obtained by the same post-processing of $\mathbf{v}$ computed by Verifier 0 in step 6. Since the prover is honest, we can write:

$$\begin{aligned}
(\mathbf{W}, \mathbf{v}) &= (\mathbf{W}, \mathbf{W}\mathbf{z}_0 + \sum_{i=1}^{S-1} \mathbf{y}_i + \mathcal{N}(0, \sigma_v^2 \mathbb{I}_k)) \\
&= (\mathbf{W}, \mathbf{W}(\mathbf{x} - \sum_{i=1}^{S-1} \mathbf{g}_i) + \sum_{i=1}^{S-1} \mathbf{y}_i + \mathcal{N}(0, \sigma_v^2 \mathbb{I}_k)) \\
&= (\mathbf{W}, \mathbf{W}(\mathbf{x} - \sum_{i=1}^{S-1} \mathbf{g}_i) + \sum_{i \in T; i \neq 0} \mathbf{y}_i + \sum_{i \notin T; i \neq 0} \mathbf{y}_i + \mathcal{N}(0, \sigma_v^2 \mathbb{I}_k)) \\
&= (\mathbf{W}, \mathbf{W}\mathbf{x} + \sum_{i \in T; i \neq 0} (\mathbf{y}_i - \mathbf{W}\mathbf{g}_i) + \sum_{i \notin T; i \neq 0} (\mathbf{y}_i - \mathbf{W}\mathbf{g}_i) + \mathcal{N}(0, \sigma_v^2 \mathbb{I}_k)).
\end{aligned}$$

Since provers outside of T follow the protocol,

$$\begin{aligned}
(\mathbf{W}, \mathbf{v}) &= (\mathbf{W}, \mathbf{W}\mathbf{x} + \sum_{i \in T; i \neq 0} (\mathbf{y}_i - \mathbf{W}\mathbf{g}_i) + \sum_{i \notin T; i \neq 0} \mathcal{N}(0, \sigma_v^2 \mathbb{I}_k) + \mathcal{N}(0, \sigma_v^2 \mathbb{I}_k)) \\
&= (\mathbf{W}, \mathbf{W}\mathbf{x} + \sum_{i \in T; i \neq 0} (\mathbf{y}_i - \mathbf{W}\mathbf{g}_i) + \mathcal{N}(0, (S - |T|)\sigma_v^2 \mathbb{I}_k)) \\
&= (\mathbf{W}, (\mathbf{W}\mathbf{x} + \mathcal{N}(0, (S - |T|)\sigma_v^2 \mathbb{I}_k)) + \sum_{i \in T; i \neq 0} (\mathbf{y}_i - \mathbf{W}\mathbf{g}_i)) \\
&\approx_{(\varepsilon, \delta)} (\mathbf{W}, \mathcal{N}(0, (S - |T|)\sigma_v^2 \mathbb{I}_k) + \sum_{i \in T; i \neq 0} (\mathbf{y}_i - \mathbf{W}\mathbf{g}_i)) \\
&= (\mathbf{W}, \mathbf{v}_{Sim}).
\end{aligned}$$

Here we have used Corollary 11 in the second to last step. ◀

The honest prover assumption is necessary to give privacy to the prover. The assumption on Verifier 0 being honest is necessary as well in the protocol as stated: a malicious Verifier 0 that can choose an adversarial $\mathbf{W}$ can violate the privacy constraint. For example, a verifier

that knows that the true $\mathbf{x}$ lies in a certain k-dimensional subspace can choose the projection matrix $\mathbf{W}$ to project to that subspace. This will make the projected vector to have length much larger than 1, and invalidate the assumptions in Lemma 8. We next show that this is the only place where we need Verifier 0 to be honest. Thus given a distributed oracle for randomly selecting $\mathbf{W}$, e.g. using shared randomness, we have privacy as long as one of the Verifiers is honest.

▶ **Theorem 16** (DZK assuming randomly chosen $\mathbf{W}$). *Suppose that* $\|\mathbf{x}\|_2 \leq 1$. *Further suppose that the prover is honest and the matrix* $\mathbf{W}$ *shared in Step 4 by Verifier* 0 *is uniformly random. Then for any* $T \subsetneq [S]$, *T's view is* $(\varepsilon + \varepsilon', \delta + \delta')$*-DZK as long as* $\sigma_v \geq 2c_\delta\sqrt{\ln \frac{4}{\delta}}/\varepsilon$ *and* $\sigma_{SS} \geq 2\sqrt{\ln \frac{2}{\delta'}}/\varepsilon'$.

Proof. The proof is nearly identical to the previous proof. When $0 \notin T$, the theorem follows from Theorem 15. When Verifier 0 is in the set, the secret sharing itself is (ε', δ')-DZK, by Theorem 12. The rest of the protocol is (ε, δ)-DZK by repeating the proof of Theorem 15. The result follows. ◀

7 Application to Robust Secure Aggregation

Our protocol for robust secure aggregation (Algorithm 5) builds on the additive secret shares with norm bound verification. The prover part of the protocol is nearly identical to secret sharing, with the only change being that the client sends its identifier j with all the shares. We assume that each client has a unique identifier, though this assumption can be easily relaxed by having the client send a random nonce instead of its identifier.

The verifiers execute the norm verification protocol for each client. Verifier 0 constructs the set of indices J^* that pass the norm verification and shares it with all the verifiers. The verifiers optionally check that J^* is large enough; this part is not needed for our summation protocol, but can be useful to ensuring that the sum itself is differentially private. The verifiers now add up the secret shares for the provers in J^* and share the sum with Verifier 0, that adds up the sums of secret shares to derive the sum.

Algorithm 5 Client Protocol for Robust Secure Aggregation.

Input: Prover j has a vector $\mathbf{x}_j \in \mathbb{R}^d$
Output: Verifiers compute $\sum_j \mathbf{x}_j$

Prover$_j(\mathbf{x}_j)$**:**
  Input: Vector $\mathbf{x}_j \in \mathbb{R}^d$ with $\|\mathbf{x}_j\| \leq 1$.
  Parameters: $\sigma_{SS} \in \mathbb{R}$.
  Generate $\mathbf{g}_1, \ldots, \mathbf{g}_{S-1} \sim \mathcal{N}(\mathbf{0}, \sigma_{SS}^2 \mathbb{I}_d)$ using private randomness.
  Send $\mathbf{x}_j - \sum_{i=1}^{S-1} \mathbf{g}_i$ to Verifier 0.
  for $i = 1 \ldots S-1$ **do**
    Send $(j, \mathbf{g}_i)$ to Verifier i.

The privacy proof is nearly identical to the last section. Indeed up to the computation of J^*, the protocol is exactly equivalent to the norm verification protocol. Verifiers other than verifier 0 do not receive any additional message after J^*, so that a simulator for a subset of verifiers excluding verifier 0 is essentially identical to that in the previous section. Verifier 0 receives a set of vectors $\{s_i\}$. For $i \neq T$, the simulator simulates $s_i \sim \mathcal{N}(\mathbf{0}, |J^*|\sigma_{SS}^2 \mathbb{I}_d)$ subject to the sum of all s_i's being equal to the output. It can be easily verified that this part of the simulation is exact. Privacy follows.

Algorithm 6 Server Protocol for Robust Secure Aggregation.

Input: Prover j has a vector $\mathbf{x}_j \in \mathbb{R}^d$
Output: Verifiers compute $\sum_j \mathbf{x}_j$

Verifier-0:
- **Parameters:** Integer k. Threshold $\tau \in \mathbb{R}$.
- Receive $V_0 = \{(j, \mathbf{z}_0^j)\}$ from Provers. Let $J_0 = \{j : (j, \mathbf{z}_0^j) \in V_0\}$.
- Generate $\mathbf{W} \in \mathbb{R}^{k \times d}$ with each $W_{ij} \sim \mathcal{N}(0, \frac{1}{k})$ using private randomness.
- Send $\mathbf{W}$ to Verifiers $1, \ldots, S-1$.
- **for** $i = 1, \ldots, S-1$ **do**
  - Receive $V_i = \{(j, \mathbf{y}_i^j)\}$ from Verifier i. Let $J_i = \{j : (j, \mathbf{y}_i^j) \in V_i\}$.
- Let $J = \cap_i J_i$.
- **for** $j \in J$ **do**
  - Compute $\mathbf{v}^j = \mathbf{W}\mathbf{z}_0^j + \sum_{i=1}^{S-1} \mathbf{y}_i^j + \mathcal{N}(\mathbf{0}, \sigma_v^2 \mathbb{I}_k)$.
  - **if** $|\mathbf{v}^j| \le \tau$ **then**
    - add j to J^*; `// J* collects j that pass the norm verification.`
- Send J^* to Verifiers $1, \ldots, S-1$.
- *Optional:* **if** *not* *Valid*(J^*) **then**
  - Abort; `// Ensure J* is large enough.`
- $\mathbf{s}_0 = \mathbf{0}$.
- **for** $j \in J^*$ **do**
  - $\mathbf{s}_0 = \mathbf{s}_0 + \mathbf{z}_0^j$.
- **for** $i = 1, \ldots, S-1$ **do**
  - Receive $\mathbf{s}_i$ from Verifier i.
- Return $\sum_{i=0}^{S-1} \mathbf{s}_i$.

Verifier-i ($i \ge 1$):
- **Parameters:** Integer k. Noise scale $\sigma_v \in \mathbb{R}$.
- Receive $V_i = \{j, \mathbf{z}_i^j)\}$ from Provers. Let $J_i = \{j : (j, \mathbf{z}_i^j) \in V_i\}$.
- Receive $\mathbf{W} \in \mathbb{R}^{k \times d}$ from Verifier-0.
- **for** $j \in J_i$ **do**
  - Compute $\mathbf{y}_i^j = \mathbf{W}\mathbf{z}_i^j + \mathcal{N}(\mathbf{0}, \sigma_v^2 \mathbb{I}_k)$.
- Send $\{(j, \mathbf{y}_i^j)\}$ to Verifier-0.
- Receive J^* from Verifier-0.
- *Optional:* **if** *not* *Valid*(J^*) **then**
  - Abort; `// Ensure J* is large enough.`
- $\mathbf{s}_i = \mathbf{0}$.
- **for** $j \in J^*$ **do**
  - $\mathbf{s}_i = \mathbf{s}_i + \mathbf{z}_i^j$.
- Send $\mathbf{s}_i$ to Verifier-0.

We next prove the correctness. We wish to prove that when all the parties are honest, then the sum is correctly computed except with a small failure probability. With probability $1 - n\beta$, each of the n norm verification steps succeed, so that J^* is the set of all clients. Conditioned on this, the correctness of the secret sharing and the commutativity of addition immediately imply that the sum computed by Verifier 0 is the desired sum of all vectors.

We note that for many applications such as gradient accumulation, a weaker correctness notion may suffice. If J^* is a random subset of $[n]$ with each j landing in J^* with probability $(1 - \beta)$, we get an unbiased estimate of the sum. For this weaker definition of correctness, the failure probability does not need to be scaled by a multiplicative factor of n which translates to a smaller threshold τ, and thus better robustness.

Finally we argue robustness. Consider a client j. If the client secret-shares a vector with norm at most ρ, then their affect on the computed sum is clearly at most ρ. On the other hand, if client j's shares add up to a vector of norm larger than ρ, it will be rejected by the norm verification step except with probability β. This means that $j \notin J^*$ and j's secret shares do not contribute at all to the compute sum. Additionally, if j does not send messages to all the verifiers, their input gets rejected as well.

When the validity check on J^* is added, the robustness claim is weaker. Indeed suppose that the validity check compares $|J^*|$ to a threshold, say $\frac{n}{2}$. Then the $(\frac{n}{2}+1)$th malicious client can cause the computation to abort. The robustness guarantee now says that if the computation succeeds, then the effect of any potentially malicious client is bounded. Further, we can argue that a small number of malicious clients cannot cause the computation to abort, except with small probability.

We have thus established correctness, robustness and privacy of our protocol. For n clients sending vectors in $\mathbb{R}^d$, the communication cost for each client is $O(d|S|)$. The communication cost between servers is $O(dk + nk + d|S|)$. Recall that a $k = O(\sqrt{\ln n})$ suffices to get polynomially small completeness and soundness.

On the Privacy of the Sum

We established the privacy of the protocol, conditioned on the sum. How do we ensure the privacy of the sum itself? One option is to add differential privacy noise to the sum itself to ensure privacy. If each verifier adds noise to s_i, we get a differential privacy guarantee against any strict subset of the verifiers. The eventual noise variance for the sum then scales with the number of servers.

An appealing alternative is to distribute the noise generation itself. This approach goes back to Dwork et al. [15]. The question of generating noise on different clients such that the sum has a certain distribution has been studied for this reason. While Gaussian noise has the nice property that sum of gaussians is a gaussian, Laplace noise is also "divisible" [21, 3]. These arguments however require that the summation be done over real numbers. In particular, this means that for privacy to hold, the constituents of the sum may need to be communicated to sufficiently high precision even if the original vectors are $\{0, 1\}$. Works such as [1] address this question of preserving privacy while reducing the communication.

Recent results on privacy amplification by shuffling offer an elegant way out of this cononudrum. The general results in this direction [7, 13, 17, 4, 18] say that local randomizers, when shuffled give strong central differential privacy guarantees. In particular, since summation is a post-processing of shuffling, these results apply to the sum. The privacy-accuracy trade-offs of the shuffle model are very competitive with the central model for many settings [30, 18]. Moreover, in deployments where local randomizers are used for other reasons, this approach avoids adding additional noise.

This ability to post-process without hurting privacy offers additional benefits. The secret-shares themselves can be rounded, truncated, or compressed without hurting privacy. For example, when the input vectors are $\{0, 1\}$, the secret sharing algorithm can use discrete gaussian noise [10], and truncate all secret shares to $[-B, B]$ for a suitable constant B. This does not affect the privacy claim, and the truncation operator is the identity except with a small probability depending on B. The small loss in accuracy due to rare truncation can be analytically or empirically traded-off against the communication cost. As an example $B = 127$ would suffice for encoding each bit as 8 bits, and would ensure that the likelihood of any single bit being distorted, say for $\sigma_{SS} = 20$ is at most 10^{-8}. This may be an acceptable

error rate in applications where randomized response is used to generate the bit vectors. In comparison the field size in PRIO must grow with the number of clients and for typical values, one would use at least 32 bits.

References

1 Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. cpsgd: Communication-efficient and differentially-private distributed sgd. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7564–7575. Curran Associates, Inc., 2018. URL: `http://papers.nips.cc/paper/7984-cpsgd-communication-efficient-and-differentially-private-distributed-sgd.pdf`.

2 Michael Backes, Aniket Kate, Sebastian Meiser, and Tim Ruffing. Secrecy without perfect randomness: Cryptography with (bounded) weak sources. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *Applied Cryptography and Network Security*, pages 675–695, Cham, 2015. Springer International Publishing.

3 B. Balle, J. Bell, A. Gascón, and Kobbi Nissim. Private summation in the multi-message shuffle model. *ArXiv*, abs/2002.00817, 2020.

4 Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 638–667, Cham, 2019. Springer International Publishing.

5 Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, pages 451–468, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

6 James Bell, K. A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly)logarithmic overhead. Cryptology ePrint Archive, Report 2020/704, 2020. URL: `https://eprint.iacr.org/2020/704`.

7 Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 441–459, New York, NY, USA, 2017. Association for Computing Machinery. `doi:10.1145/3132747.3132769`.

8 Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery. `doi:10.1145/3133956.3133982`.

9 Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear pcps. Cryptology ePrint Archive, Report 2019/188, 2019. URL: `https://ia.cr/2019/188`.

10 Clément L. Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy, 2020. `arXiv:2004.00010`.

11 T. H. Hubert Chan, Elaine Shi, and Dawn Song. Privacy-preserving stream aggregation with fault tolerance. In Angelos D. Keromytis, editor, *Financial Cryptography and Data Security*, pages 200–214, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

12 Albert Cheu, Adam Smith, and Jonathan Ullman. Manipulation attacks in local differential privacy, 2019. `arXiv:1909.09630`.

13 Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 375–403, Cham, 2019. Springer International Publishing.

14 Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 259–282, Boston, MA, 2017. USENIX Association. URL: `https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/corrigan-gibbs`.

15 Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology (EUROCRYPT 2006)*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer Verlag, May 2006. URL: `https://www.microsoft.com/en-us/research/publication/our-data-ourselves-privacy-via-distributed-noise-generation/`.

16 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014.

17 Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 2468–2479, USA, 2019. Society for Industrial and Applied Mathematics.

18 Vitaly Feldman, Audra McMillan, and Kunal Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2021. arXiv:2012.12803 [cs.LG].

19 Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Amer Sinha. Differentially private aggregation in the shuffle model: Almost central accuracy in almost a single message. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3692–3701. PMLR, 18–24 July 2021. URL: `https://proceedings.mlr.press/v139/ghazi21a.html`.

20 Badih Ghazi, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Private aggregation from fewer anonymous messages. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 798–827, Cham, 2020. Springer International Publishing.

21 Slawomir Goryczka and Li Xiong. A comprehensive comparison of multiparty secure additions with differential privacy. *IEEE Transactions on Dependable and Secure Computing*, 14:463–477, 2017.

22 Vipul Goyal, Ilya Mironov, Omkant Pandey, and Amit Sahai. Accuracy-privacy tradeoffs for two-party differentially private protocols. In *CRYPTO*, pages 298–315. Springer, 2013. `doi:10.1007/978-3-642-40041-4_17`.

23 Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography from anonymity. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 239–248, 2006.

24 P. Kairouz, S. Oh, and P. Viswanath. Differentially private multi-party computation. In *2016 Annual Conference on Information Science and Systems (CISS)*, pages 128–132, March 2016. `doi:10.1109/CISS.2016.7460489`.

25 Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Secure multi-party differential privacy. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2008–2016. Curran Associates, Inc., 2015. URL: `http://papers.nips.cc/paper/6004-secure-multi-party-differential-privacy.pdf`.

26 B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *Ann. Statist.*, 28(5):1302–1338, October 2000. `doi:10.1214/aos/1015957395`.

27 Andrew McGregor, Ilya Mironov, Toniann Pitassi, Omer Reingold, Kunal Talwar, and Salil Vadhan. The limits of two-party differential privacy. In *51st Annual Symposium on Foundations of Computer Science*, pages 81–90. IEEE, 2010.

28 Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 126–142, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

29 Jinhyun So, Basak Guler, and A. Salman Avestimehr. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning, 2020. arXiv:2002.04156.
30 Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation, 2020. arXiv:2001.03618.